UNITED STATES PATENT APPLICATION

for

METHOD AND SYSTEM FOR DESIGNING OBJECTS USING FUNCTIONAL OBJECT REPRESENTATION

Inventors:

Attilio Rimoldi
Gian Paolo Bassi

prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Los Angeles, CA 90025-1026
(408) 720-8300

File No.: 05826.P002

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: EL431887895US

Date of Deposit: ______

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner of Patents and Trademarks, Washington, D. C. 20231

Michelle Begay
(Typed or printed name of person mailing paper or fee)

Michelle Begay
(Signature of person mailing paper or fee)

February 8, 2002
(Date signed)

METHOD AND SYSTEM FOR DESIGNING OBJECTS USING FUNCTIONAL OBJECT REPRESENTATION

RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 60/323,193, filed September 17, 2001.

FIELD OF THE INVENTION

[0002] The present invention relates to the field of computer-aided design (CAD) of physical parts and assemblies; more particularly, the present invention relates to designing objects using functional object representation.

BACKGROUND

[0003] A typical computer-aided design (CAD) system employed in engineering contexts uses a geometry-oriented approach to define and represent engineering information. As a result, a designer is required to perform a number of low-level geometric operations in order to produce a final digital model of a desired product. In other words, the designer has to focus on the details of geometry creation rather than the required functionality of the product.

[0004] In addition, conventional CAD systems use a rigid history-based modeling approach that creates dependency between the operations performed by the designer when creating a digital model. That is, an earlier operation may influence subsequent operations or, alternatively, a subsequent operation may

obliterate the functionality of a prior operation. This order dependent nature of feature operations makes it difficult for the designer to apply any modification to the digital model or correct mistakes occurred early in the design process.

[0005] Further, conventional CAD systems do not allow members of a product development team to work together simultaneously, regardless of their location or role, in order to create and finalize product definitions. Instead, current design intent collaborative practices are based on serial collaboration, in which copies of digital product files are passed back and forth among development team members who must wait to get these copies back before they can make or even suggest design refinements.

SUMMARY OF THE INVENTION

[0006] A method and apparatus for designing objects using functional object representation are described. According to one aspect of the present invention, the method includes receiving behavioral parameters associated with each design feature of an object from a user and determining a correspondence between each design feature and one or more body partitions from a predefined set of body partitions that compose the body of the object. The determination is made based on the behavioral parameters received from the user. The method further includes assigning a contributing volume to each of the corresponding body partitions and computing the digital model of the object using contributing volumes of all design features of the object.

[0007] Other features of the present invention will be apparent from the accompanying drawings and from the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention, which, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only.

[0009] **Figure 1** is a block diagram of one embodiment of a tool for designing objects using functional modeling.

[0010] **Figure 2** is a flow diagram of one embodiment of a process for designing objects using functional modeling.

[0011] **Figures 3 and 4** illustrate examples of user interfaces that can be used in one embodiment of the functional modeling process.

[0012] **Figure 5** is a block diagram of one embodiment of a system 500 for designing objects using functional object representation (FOR).

[0013] **Figure 6** is a flow diagram of one embodiment of a process for designing objects using functional object representation (FOR).

[0014] **Figure 7** illustrates one embodiment of a static Boolean tree that can be used to compute a digital model.

[0015] **Figures 8A-8O** illustrate various examples of design features and volume types created by the design features.

[0016] **Figure 9** illustrates one embodiment of a partial process of generating data pertaining to a digital model of an object using a static Boolean tree.

[0017] **Figure 10** is block diagram of one embodiment of a network architecture used for design collaboration.

[0018] **Figure 11** is a block diagram of one embodiment of a client module 1012 of **Figure 10.**

[0019] **Figure 12** is a flow diagram of one embodiment of a process for designing objects using design intent merge.

[0020] **Figure 13** is a block diagram of an exemplary computer system.

DETAILED DESCRIPTION

[0021] A method and system for designing objects using functional object representation are described. In the following description, numerous details are set forth. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

[0022] Some portions of the detailed descriptions that follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0023] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as

apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0024] The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0025] The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the

required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

[0026] A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory ("ROM"); random access memory ("RAM"); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

Overview

[0027] A mechanism for designing objects using functional modeling is described. Unlike conventional designing tools that require designers to focus on low level geometric details, functional modeling allows the designers to concentrate on behavioral parameters of design elements. Behavioral parameter defines the required engineering functionality of design elements.

In one embodiment, a user creating a digital model of an object is provided with a tool that allows the user to choose a design element (also referred to herein as a design feature) needed for this object. For example, the user may choose a pocket, a cutout, a rib, a boss, a box, a cylinder, etc. The set of design features offered for the

user's selection may vary depending on the industry for which the object is to be used. Further, for each design feature, the user is provided with one or more behavioral parameters that define the functionality of the design feature. For example, for a pocket, a behavioral parameter may define the thickness of the pocket's walls. In one embodiment, behavioral parameters of design features are defined based on the rules and logic of a specific application in which these design features are to be used.

[0028] The user can select some specific behavioral parameters that reflect the functionality intended by the user for this design feature. In one embodiment, behavioral parameters specified by the user also define the relations between this design feature and other design elements associated with the object. For example, the user creating a hole in the existing body of the object may specifies that the wall thickness of the hole should be the same as the body thickness, thus requesting that the wall thickness of the hole be modified whenever the body thickness is changed.

[0029] Based on the behavioral parameters, the design feature is automatically modeled. That is, the geometry of the design feature is calculated and generated without further involvement of the user. In one embodiment, the user may also select some geometric parameters (e.g., a diameter of a hole) for the design feature. These geometric parameters are also utilized when generating the geometry of the design feature.

[0030] In one embodiment, the modeling is performed using a functional object representation (FOR) approach. Specifically, the body of an object being

modeled is presented as a predefined set of body partitions. Each body partition is associated with a certain volume type. These volume types may include, for example, a volume representing a core of the object, a volume representing a cavity inside the object, a volume added outside of the core, a volume added across the core, a volume representing a clearance area within the object (e.g., a cutout), etc.

[0031] When the user selects a certain combination of behavioral parameters for a design feature, this combination is analyzed to determine which body partitions correspond to the design feature. The corresponding body partitions are then assigned volumes created by the design feature. These operations are performed for each design feature associated with the object. As a result, depending on a particular design, some body partitions may have volumes contributed by one or more design features while others may have no contributing volumes.

[0032] In one embodiment, for each digital model, a predefined set of body partitions is stored in a database with corresponding contributing volumes. Each body partition that has a contributing volume may also store a link to a design feature that has created this contributing volume. This link provides access to data pertaining to this design feature, including behavioral and geometrical parameters of the design feature. Accordingly, for every digital model, the database stores a set of data that encapsulates the behavioral properties of each design feature associated with the object.

[0033] Further, the contributing volumes of each design feature associated with the object are utilized to compute the digital model of the object. In one

embodiment, a constant algorithm is used for the computation. That is, the algorithm may be defined and used for all objects in any industry or, alternatively, for all objects in a specific industry or application. In one embodiment, the digital model of the object can be re-computed at any time using the set of data stored in the database.

[0034] Accordingly, functional modeling abstracts the design features from their geometrical representation and allows the user to focus on high-level functional elements and operations, rather than the creation of the detailed geometry. In addition, by maintaining a set of data that encapsulates the behavioral properties of each design feature associated with the object during the design process, a history-independent modeling approach is provided, in which each design feature retains its functionality throughout the design process, regardless of any subsequent or prior modeling operations related to the object. Furthermore, functional modeling automatically infers and maintains the relationships between the design features of the digital model during the entire design process.

[0035] Functional modeling can also be used to allow simultaneous development of an object by several designers. That is, once a first designer makes a change to a digital model, the first designer can submit a request to communicate this change to a second designer. In one embodiment, this request causes a design tool operated by the first designer to extract data pertaining to the change from the database and send it to a server which forwards this data to a client device of the second designer.

[0036] Once a design tool residing on the client device of the second designer receives the data pertaining to the change, it analyzes this data, together with data pertaining to a digital version model created by the second designer, and integrates the change performed by the first designer into the digital model version created by the second designer. The same process can also be performed if the second designer modifies the digital model and decides to communicate this modification to the first designer. This process can be performed for any number of designers participating in simultaneous object development. As a result, multiple participants in object development can work in parallel, regardless of their location and role, to create and finalize object definitions.

Functional Modeling

[0037] **Figure 1** is a block diagram of one embodiment of a tool 100 for designing objects using functional modeling. Referring to **Figure 1**, functional modeling tool 100 includes a parameter controller 106, a user interface 102, a modeler 104 and a database 108. Parameter controller 106 maintains behavioral parameters for various design features. The design features may include a pocket, a cutout, a rib, a boss, a box, a hole, etc. The behavioral parameters define the functionality of design features. For example, a behavioral parameter may pertain to the shellability of a box or the thickness of a pocket's wall. In one embodiment, behavioral parameters of design features are defined based on the rules and logic of a specific application in which these design features are used.

[0038] User interface 102 displays a set of features to the user and further displays a list of behavioral parameters for a feature selected by the user. The user can then select one or more behavioral parameters from the list according to the functionality intended by the user for this design feature. In one embodiment, the user may also select some geometric parameters (e.g., a diameter of a hole) for the design feature. In one embodiment, behavioral parameters specified by the user not only define the functionality of the design feature itself but also the relations between this design feature and other design elements associated with the object. In one embodiment, the behavioral parameters specified by the user as well as other data pertaining to the design feature are stored in database 108.

[0039] Modeler 104 is responsible for modeling design feature upon receiving data pertaining to the user's selection of behavioral parameters (and geometric parameters when specified by the user). That is, modeling algorithm 104 calculates and generates the geometry of the design feature without user involvement. The calculation is performed using the behavioral parameters specified by the user. In one embodiment, the calculation is performed using a constant algorithm as will be described in more detail below.

[0040] In one embodiment, the user is allowed to modify the parameters of the design feature at any time during the design process. If the parameters being modified are geometric, the new geometry of the design feature retains the functionality that was intended by the user initially. Alternatively, if the user

requests to modify any behavioral parameter of the design feature, the new geometry of the design feature will reflect the change to the feature's functionality.

[0041] In one embodiment, a change of one design feature may result in an automatic change of another design feature to maintain the intended functionality of the other design feature.

[0042] Accordingly, the design features created by modeling algorithm 104 are not affected by any previous or subsequent modeling operations unless specifically intended by the user. These design features may be associated into semantically complete clusters that can be used in different contexts.

[0043] **Figure 2** is a flow diagram of one embodiment of a process 200 for designing objects using functional modeling. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

[0044] Referring to **Figure 2,** process 200 begins with processing logic providing a set of behavioral parameters associated with a design feature selected by a user (processing block 204). In one embodiment, a user interface is provided to display the set of behavioral parameters and facilitate the user's selection of these behavioral parameters as will be described in more detail below in conjunction with **Figure 3 and 4.**

[0045] At processing block 206, processing logic receives one or more behavioral parameters specified the user. In one embodiment, processing logic also receives some geometric parameters specified by the user.

[0046] At processing block 208, processing logic models the design feature based on the behavioral parameters specified by the user. In one embodiment, the modeling is performed using a functional object representation approach as will be described in greater detail below.

[0047] **Figures 3 and 4** illustrate examples of user interfaces that can be used in one embodiment of the functional modeling process.

[0048] Referring to **Figure 3**, a user interface 300 displays a set 320 of design features including an extrude feature selected by the user. A set of parameters available for the extrude feature includes a volume type parameter 306, a solidity parameter 304 and a shellability parameter 302. Parameter 308 is a geometric parameter that defines the thickness of the shell.

[0049] Design elements 310, 312, 314 and 316 were each added to object 318 using various behavioral parameters associated with the extrusion feature. Design element 310 was created by selecting "solid" for parameter 304, "non-shellable" for parameter 302 and "default" for parameter 306. The "default" volume type represents a volume added outside of the existing body.

[0050] Design element 312 was created by selecting "solid" for parameter 304, "shellable" for parameter 302 and "default" for parameter 306. Because the shellable

parameter was selected, the cavities of design element 310 and existing design element 318 are integrated together.

[0051] Design element 314 was created by selecting "solid" for parameter 304, "non-shellable" for parameter 302 and "add" for parameter 306. The "add" volume type represents a volume added both inside and outside of the existing body.

[0052] Design element 316 was created by selecting "solid" for parameter 304, "non-shellable" for parameter 302 and "subtract" for parameter 306. The "subtract" volume type represents a volume subtracted from the existing body.

[0053] Referring to **Figure 4**, user interface 400 displays a set 408 of design features including a cutout feature selected by the user. A set of parameters available for the cutout feature includes a hole parameter 404 (specifying whether the cutout has a bottom) and a wall thickness parameter 406. Parameter 402 is a geometric parameter that defines the length of the cutout.

[0054] Cutouts 412, 314 and 416 were each created using various behavioral parameters associated with the cutout feature. Cutout 412 was created by selecting a "use body thickness" option for parameter 406. As a result, the wall thickness of cutout 412 will be maintained the same as that of the body.

[0055] Cutout 414 was created by selecting a "no wall" option for parameter 406. Accordingly, cutout 414 was created without the wall.

[0056] Cutout 416 was created by selecting an "enter thickness" option for parameter 406 and specifying a particular size. Because this size is larger than that of the body, the wall of cutout 416 is thicker than the rest of the walls.

Functional Object Representation

[0057] Functional object representation (FOR) provides a history-independent approach for creating and maintaining data pertaining to a digital model. The FOR is based on a predefined set of body partitions that compose the body of the object being modeled. Each body partition is associated with a certain volume type. These volume types may include, for example, a volume representing a core of the object, a volume representing a cavity inside the object, a volume added outside of the core, a volume added both inside and outside of the core, a volume representing a clearance area within the object (e.g., a cutout), etc.

[0058] **Figure 5** is a block diagram of one embodiment of a system 500 for designing objects using functional object representation (FOR). Referring to **Figure 5**, system 500 includes a modeler 512 and a database 508. In one embodiment, modeler 512 includes a user input processor 502, a volume distributor 504, and a model generator 506.

[0059] User input processor 502 is responsible for receiving behavioral parameters selected by the user for each design feature and determining a correspondence between the behavioral parameters and body partitions. In one embodiment, a correspondence between a body partition and one or more

behavioral parameters exists if a design feature creates a volume type associated with this body partition.

[0060] Volume distributor 504 is responsible for assigning volumes created by each design feature to corresponding body partitions. Depending on the design of the object being modeled, some body partitions may have volumes from one or more design features while others may have no contributing volumes.

[0061] Model generator 506 is responsible for computing a digital model of the object using the contributing volumes of each design feature associated with the object. In one embodiment, a constant algorithm is used for the computation. That is, this algorithm may be defined and used for all objects in any industry or, alternatively, for all objects in a specific industry or application. In one embodiment, the algorithm is defined by a static Boolean tree as will be described in greater detail below.

[0062] In one embodiment, database 508 stores a predefined set of body partitions with corresponding contributing volumes for each digital model. Each body partition that has a contributing volume may also store a link to a design feature that has created this contributing volume. This link provides access to data pertaining to this design feature, including behavioral and geometrical parameters of the design feature. Data pertaining to design features may also be stored in database 508.

[0063] Model generator 506 can re-compute the digital model of the object at any time using the data stored in database 508.

[0064] Figure 6 is a flow diagram of one embodiment of a process 600 for designing objects using functional object representation (FOR). The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

[0065] Referring to **Figure 6**, process 600 begins with processing logic receiving behavioral parameters associated with each design feature of an object from the user (processing block 604). At processing block 606, processing logic determines a correspondence between each design feature and one or more body partitions using the behavioral parameters. That is, based on its specified behavioral parameters, a design feature may contribute volumes to certain body partitions. Processing logic analyzes the behavioral parameters specified for each design feature and determines which body partitions should receive contributing volumes.

[0066] At processing block 608, processing logic assigns contributing volumes to corresponding body partitions. In one embodiment, the predefined set of body partitions and corresponding contributing volumes are stored in the database, together with links to data pertaining to the design features. This data may include a set of behavioral and geometrical parameters associated with each design feature. In one embodiment, the database also stores links between related design features of the object.

[0067] Afterwards, at processing block 610, processing logic computes the digital model of the object using contributing volumes of each design feature of the

object. Subsequently, the user may request to modify an existing design feature or add a new design feature. Then, processing logic will reassign the contributing volumes according to this change and re-compute the digital model of the object. Accordingly, a designer can implement and change design elements at any stage in the design process, regardless of the order of operation.

[0068] **Figure 7** illustrates one embodiment of a static Boolean tree that can be used to compute a digital model. Referring to **Figure 7**, a body of an object is presented as a combination of body partitions. Each body partition is associated with a certain volume type. The volume types include shellable volumes (SV), positive volumes (PV), negative volumes (NV), negative volumes subtracted from a cavity within the body (NV*), operational shellable volumes (OSV), functional positive volumes (FPV), and functional negative volumes (FNV).

[0069] SV represents the volume of the body' core. OSV represents the volume of the body's cavity. PV represents positive volumes added outside of the body. NV represents negative volumes subtracted from volumes located outside of the body. NV* represents negative volumes subtracted from the cavity. FPV represents positive volumes added both inside and outside of the core. FNV represents negative volumes of clearance areas inside and outside of the body.

[0070] **Figures 8A-8O** illustrate various examples of design features. For each design feature, one or more volume types associated with this feature are identified.

[0071] Referring back to **Figure 7**, the static Boolean tree defines the following constant algorithm for computing a digital model:

[0072] Final body = $SV + PV - [OSV + (NV - SV) - NV^*] + FPV - FNV$.

This algorithm, which may be used for computing a digital model of any object or any object in a specific industry, is history-independent because it makes the order of introduction of the above volumes irrelevant for the resulting digital model.

[0073] **Figure 9** illustrates one embodiment of a partial process of generating data pertaining to a digital model of an object using a static Boolean tree.

[0074] Referring to **Figure 9**, body 908 of an object consisting of a box 902 and a hole 904 is created. As shown, box 902 contributes volume to body partition 910 associated with FPV and hole 904 contributes volume to body partition 906 associated with FNV. Body partitions 906 and 910 are linked to data 912 and 914 that identify corresponding design features. Data 912 and 914 are in turn linked to parameters 916 and 918 (behavioral and geometrical) of corresponding design features. Contributing volumes FNV and FPV are calculated using the geometrical parameters of corresponding design features.

[0075] In one embodiment (not shown), a product consists of two or more related objects. In this embodiment, the body of one object may be linked to the body of a related object.

[0076] The data structure discussed above only requires to capture the functional data of individual design elements, thus providing an efficient way of storing and manipulating a set of data required for generating the digital model. Accordingly, this set of data can be easily transported via a network.

Design Intent Merge

[0077] Figure 10 is a block diagram of one embodiment of network architecture used for design collaboration.

[0078] Referring to **Figure 10**, clients 1006 represent computers of each participant in simultaneous object development. Each client 1006 is connected to a server 1004 via a network 1002. Network 1002 may be any private network (e.g., LAN) or public network (e.g., Internet, wireless network, etc.).

[0079] Each client 1006 includes a client module 1012 which is responsible for packaging data and transmitting it to server 1004 in response to a request of a participant to share his or her modification of the digital model with other participants of simultaneous product development.

[0080] Server 1004 includes a server module 1008 which operates like a directory service, receiving data from one or more clients 1006 and sending, when directed, the data to other clients 1006 for a merge with other participants' data.

[0081] Server 1004 also includes a database 1010 which stores data pertaining to digital models created by users of clients 1006. In one embodiment, database 1010 stores data pertaining to different versions of the digital model, showing each participant's contribution into design features and associations between design features. In one embodiment, database 101 also maintains a persistent identity of each modification by storing information identifying the location and the time of each modification.

[0082] Participants can work in a synchronous mode, i.e., work together in real time, constantly sharing and merging their modeling operations. Alternatively, the participants can work in an asynchronous mode, i.e., work on their own tasks offline, and share, receive and merge modeling operations whenever they choose to go online.

[0083] **Figure 11** is a block diagram of one embodiment of a client module 1012 of **Figure 10**. Referring to **Figure 11**, client module 1012 includes a data receiver 1104, a data merger 1106, a data supplier 1102, and a database 1108.

[0084] Data supplier 1102 is responsible for processing a participant's request to share a change he or she made to a digital model with another participant or participants. In one embodiment, data supplier 1102, upon receiving the participant's request, extracts data pertaining to the change from database 1108 and sends the extracted data to server 1004, with information identifying a recipient participant(s). In one embodiment, data supplier 1102 automatically extracts all data related to a modified design feature.

[0085] In one embodiment, the participant may request to protect the design feature being sent from other participants' changes to this design feature. Data supplier 1102 includes information reflecting this limitation in the packet sent to the server. In one embodiment, the participant is also provided with an option to send all recent changes (i.e., the changes made after the last transmission), rather than a specific change to the recipients. Data supplier 1102 determines what changes have

been made since the last transmission, extracts data pertaining to these changes and sends it to server 1004.

[0086] Data receiver 1104 is responsible for receiving changes made by other participants. Data merger 1106 is responsible for analyzing data received from other participants and data pertaining to the version created by this participant, and based on this analysis, integrating the changes performed by the other participants into the version created by this participant. For example, data merger 1106 may receive an identifier of a modified design feature, contributing volumes created by the modified design feature and geometric parameters of the modified design feature. Data merger 1106 then retrieves corresponding data from database 1108 using the design feature's identifier, compares the contributing volumes and geometric parameters of the modified design feature with corresponding data stored in database 1108, and updates database 1108 with the received information. **Figure 12** is a flow diagram of one embodiment of a process 1200 for designing objects using design intent merge. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both.

[0087] Referring to **Figure 12**, process 1200 begins with processing logic receiving data concerning a change made by a first participant over a network (processing block 1204). At processing block 1206, processing logic analyzes the data concerning the change made by the first participant and data concerning a digital model version created by a second participant. At processing block 1208, processing

logic integrates the change performed by the first participant into the digital model version created by the second participant based on the analysis performed at processing block 1206, as described in more detail above.

[0088] In one embodiment, processing logic subsequently receives a request from the second participant to inform the first participant about the change made by the second participant. In response to this request, processing logic extracts data concerning the change and sends the extracted data to a server that stores it and forwards it to the first participant.

<u>An Exemplary Computer System</u>

[0089] **Figure 13** is a block diagram of an exemplary computer system 1300 that may be used to perform one or more of the operations described herein. In alternative embodiments, the machine may comprise a network router, a network switch, a network bridge, Personal Digital Assistant (PDA), a cellular telephone, a web appliance or any machine capable of executing a sequence of instructions that specify actions to be taken by that machine.

[0090] The computer system 1300 includes a processor 1302, a main memory 1304 and a static memory 1306, which communicate with each other via a bus 1308. The computer system 1300 may further include a video display unit 1310 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)). The computer system 1300 also includes an alpha-numeric input device 1312 (e.g., a keyboard), a cursor control device 1314 (e.g., a mouse), a disk drive unit 1316, a signal generation device

1320 (e.g., a speaker) and a network interface device 1322.

[0091] The disk drive unit 1316 includes a computer-readable medium 1324 on which is stored a set of instructions (i.e., software) 1326 embodying any one, or all, of the methodologies described above. The software 1326 is also shown to reside, completely or at least partially, within the main memory 1304 and/or within the processor 1302. The software 1326 may further be transmitted or received via the network interface device 1322. For the purposes of this specification, the term " computer-readable medium" shall be taken to include any medium that is capable of storing or encoding a sequence of instructions for execution by the computer and that cause the computer to perform any one of the methodologies of the present invention. The term "computer-readable medium" shall accordingly be taken to included, but not be limited to, solid-state memories, optical and magnetic disks, and carrier wave signals.

[0092] Whereas many alterations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that any particular embodiment shown and described by way of illustration is in no way intended to be considered limiting. Therefore, references to details of various embodiments are not intended to limit the scope of the claims which in themselves recite only those features regarded as essential to the invention.